Pergamon

Pattern Recognition, Vol. 28, No. 11, pp. 1751–1764, 1995
Elsevier Science Ltd
Copyright © 1995 Pattern Recognition Society
Printed in Great Britain. All rights reserved
0031-3203/95 $9.50 + .00

0031-3203(95)00035-6

ON CORRESPONDENCE, LINE TOKENS AND MISSING TOKENS§

CHIA-HOANG LEE* and ANUPAM JOSHI†‡
* Department of Computer and Information Science, National Chiao-Tung University, Hsinchu, Taiwan 30050, R.O.C.
† Department of Computer Science, Purdue University, West Lafayette, IN 47907, U.S.A.

(*Received* 8 *September* 1993; *in revised form* 22 *February* 1995; *received for publication* 17 *March* 1995)

Abstract—In this article, the authors expand on some of their previous work on correspondence and propose a system that can correspond in the presence of both point and line tokens, or a mix of them. It can also work if some tokens present in one frame are not present in the other. The proposed algorithm is an amalgamation of reductionist and holist paradigms. We establish that properties which remain invariant for whole image sequences and sequences consisting of corresponding parts from each image frame can be used to establish correspondence. Extensive computer simulation results on synthesised data as well as real image sequences are presented to validate our claims.

Correspondence Image sequence Line tokens Missing tokens Parallelism

1. INTRODUCTION

It can be said with some confidence that the "motion problem", in all its various aspects, is one of the most important components of vision. There is literature-a-plenty which deals with it. The various approaches to measurement of visual motion can be broadly categorized as either relying on optical flow techniques or on feature based techniques. It is with the latter that we concern ourselves in this work.

Feature based schemes operate by first extracting features, or tokens, from the raw images. They then establish correspondence between these features across different frames, and using these correspondences obtain not only the parameters that describe the motion in the image sequence, but also, the three dimensional (3D) structure of the objects(s) in the image. Establishing the correspondence is clearly a prerequisite to further processing in feature based schemes. Many efforts in the area of dynamic image analysis, however, assume that this underlying problem of correspondence has been resolved.(1–12) There even have been efforts to do image analysis without explicitly using point to point correspondence.(13–15) This, of course, is not to suggest that work has not been done on the issue. There is much literature devoted to the study of the correspondence problem. Salari and Sethi(16) and Jain and Sethi(17) for instance, use continuity of motion across several frames to track feature points. Other researchers have suggested keeping track of feature points(18,19) across incremental movements. Photometric attributes of an area(20) and local structure(21) have also been used as attributes to establish correspondence. In reference (22) the authors try to obtain motion parameters without using correspondence, and use that to obtain correspondence between 3D point sets. Another technique dealing with 3D points, which assumes the existence of three known matches, is proposed in reference (23). Perspective invariant descriptions of planar point sets are used for correspondence.(24) Grimson and Lozano-Perez(25) propose a tree pruning algorithm which needs exhaustive examination of various mutual relationships between the parameters they define for all token pairs. Rangarajan and Shah(26) have tried to use a proximal uniformity constrain to obtain correspondence. In reference (27) the authors use multiple attributes (like edgeness, intensity, cornerness) and multiple constraints (like intra-region smoothing, occlusion). Their method combines feature based and optical flow based schemes. Liu and Huang(28) propose a scheme for correspondence when the motion involved approximates that of a vehicle. They assume multiple frames of motion being available. Moreover, since they compute the centroid of the feature points in every frame, they assume that the same feature points are available in all frames. Three frames with lines as features are used in the algorithm proposed in reference (29) to establish correspondence. In that work, the authors comment upon the uniqueness of the closed form solution obtained, and also show how optimization techniques can be used to improve the solution in presence of noise. Weng(30) introduces Windowed Fourier Phase as a matching primitive for stereo and motion corre-

§This work was supported in part by the National Science Council of the R.O.C. under grant NSC-82-0408-E-009-366 and by NSF awards CDA-9123592 and ASC 9404859.

‡This author would like to acknowledge the support of a research fellowship from the Purdue Research Foundation. Please address all correspondence to this author.

spondence. Assuming that the data available is 3D, we have also proposed a correspondence algorithm using neural networks.[31,32]

However, most of the earlier works operate under some kinds of assumptions. Some assume availability of 3D data, some assume that there are no missing tokens in successive frames, others that the motion between frames is smooth and continuous. Most also deal only with point tokens and many need multiple frames. Still others use multiple attributes for each token, or assume orthographic projections. The search is still on, in other words, for a robust algorithm to do correspondence in presence of different kinds of tokens, especially when not all tokens present in one frame are necessarily present in the other.

In the following sections we will describe a robust algorithm for correspondence that we have proposed, which works for both point and line tokens, as also when there are missing tokens between frames. It extends an algorithm that we had proposed earlier,[33] which worked when the input had point tokens and assumed that all tokens would be present in the two frames. We will begin by briefly describing the philosophy behind the algorithm, and recapitulating its mathematical details in terms of point tokens. We will then present techniques for the algorithm to work for line tokens, as well as in the presence of missing tokens. Results of applying the algorithm to synthesised as well as real data will also be presented.

2. BASICS OF THE ALGORITHM

2.1. *Philosophical and logical basis*

Our approach to the correspondence problem is *geometric* and, in a sense *reductionist*. What we mean by reductionist in this context is that we solve the problem by using the results of computations on many subsets of the original problem. More formally, let $\mathscr{S}$ be the set of tokens obtained from two frames. Further, let $\hat{\mathscr{S}}$ be a subset of $\mathscr{S}$ that satisfies the following property: if $\hat{\mathscr{S}}$ contains a tokens from one of the frames, then it also contains its corresponding token from the other frame. We seek a property, say X, such that if X is true of $\mathscr{S}$, then it is necessarily true for all $\hat{\mathscr{S}}$, and necessarily false for any other subset. In essence, *if something is true for the whole, then it must be true for the corresponding parts.* Our approach is geometric, or rather motion based, in as much as we look for this X in the geometrical descriptors associated with the image sequence. There is psychological evidence to suggest that motion cues help in establishing correspondence.[34]

Given such a property, one could test if an arbitrary subset of tokens was of the type $\hat{\mathscr{S}}$ by testing the truth of this property. One could also use it to establish correspondence.

Claim 1. *If a property of type X exists for the token set, then it provides a necessary and sufficient condition to establish correspondence.*

Proof. Suppose we wish to determine the corresponding token in the second frame for the ith token in the first frame. If n be the number of tokens in the second frame, we could do this by forming n subsets of tokens. Each subset would consist of all the tokens except the ith token from the first frame and the jth, $(j = 1,\ldots,n)$ token from the second frame. Assuming uniqueness of correspondence, only one of these subsets would satisfy property X, and this would be a *necessary and sufficient* condition to conclude that the token k dropped from the second set to obtain this subset is the one that corresponds to the ith token of the first frame. □

Properties like X constitute what we will call 'strong" properties. We now introduce a weaker version of this property. We demand here only that if property wX is true of $\mathscr{S}$, then it be necessarily true for all $\hat{\mathscr{S}}$. We make no comment about the truth or otherwise of the property for other subsets of $\mathscr{S}$. Such properties too are useful in establishing correspondence.

Claim 2. *If a property of type wX exists, then it provides a necessary condition to establish correspondence.*

Proof. Analogous to the proof of claim 1, we construct the n subsets. It then follows that only the subsets where wX holds can be considered as candidates for establishing correspondence. Note that wX being true for a set does not guarantee that the tokens dropped to obtain it were the corresponding ones. This provides the conceptual framework for our method. □

Before moving on to the mathematical description of the problem, allow us to say that our object in this work is to establish correspondence. While we use motion parameter in order to do that, our object is not to accurately compute them. Moreover, we feel that just as in natural vision systems, the problem of correspondence can be completely solved only by combining a variety of visual cues. We only claim to provide one of the cues, albeit a cue that seems to be able to establish correspondence in most cases. We also limit ourselves in this work to establishing correspondences given that tokens have already been extracted. Consequently, we eschew any discussion of low level image processing.

2.2. *Mathematical basics*

In a previous publication,[33] the authors have described a method to obtain point correspondences between two frames assuming that no tokens are missing. For completeness, we summarise the mathematical aspects of that method here.

The problem of correspondence can be stated as follows. Given A, a set of tokens from the first frame and B, a set of tokens from the second frame, find a permutation σ such that the token A_i has the token $B_{\sigma(i)}$ as its corresponding element. In the following description, assume that the tokens are points and that from the first frame to the second, each point has undergone a rotation of $\mathbf{R}$ and a translation of $\mathbf{T}$.

Consider

$$\begin{aligned} z_iA_i &= \text{Position vector of the } i\text{th point in the first frame} \\ z'_{\sigma(i)}B_{\sigma(i)} &= \text{Position vector of the } i\text{th point in the second frame} \\ A_i &= (x_i, y_i, 1)^t \\ B_{\sigma(i)} &= (x'_{\sigma(i)}, y'_{\sigma(i)}, 1)^t \end{aligned}$$

where (x_i, y_i) and $(x'_{\sigma(i)}, y'_{\sigma(i)})$ are, respectively, the image coordinates of the ith point in the first and second frames and z_i and z'_i are their depths

Then

$$z'_{\sigma(i)}B_{\sigma(i)} = \mathbf{R}z_iA_i + \mathbf{T} \quad i = 1,\ldots,N \tag{1}$$

If we assume that the translation vector is small compared to the distance of the object and can be neglected, we can approximate equation (1) above by

$$z'_{\sigma(i)}B_{\sigma(i)} = \mathbf{R}z_iA_i \quad i = 1,\ldots,N \tag{2}$$

Using $\hat{A}_i$'s and $\hat{B}_i$'s, the unit vectors of A_i and B_i, respectively, and multiplying each side by its transpose, the above can then be rewritten as

$$\hat{B}_{\sigma(i)}\hat{B}^t_{\sigma(i)} = \mathbf{R}\hat{A}_i\hat{A}_i^tR^t \quad i = 1,\ldots,N \tag{3}$$

Let us sum the above equation over i. Since the summation is over all points, the order in which they appear makes no difference. The permutation σ is thus irrelevant and can be dropped in the following formula.

$$\sum_{i=1}^{N} \hat{B}_{\sigma(i)}\hat{B}^t_{\sigma(i)} = \sum_{i=1}^{N} \mathbf{R}\hat{A}_i\hat{A}_i^t\mathbf{R}^t \tag{4}$$

Consequently,

$$\mathbf{R}Q_1\mathbf{R}^t = Q_2 \tag{5}$$

where $Q_2 = \sum_{i=1}^N \hat{B}_i\hat{B}_i^t$, and $Q_1 = \sum_{i=1}^N \hat{A}_i\hat{A}_i^t$. This formula implies that Q_1 and Q_2 should have the same eigenvalues. We now do a SVD.

Let $Q_1 = VDV^t$ and $Q_2 = UDU^t$. It follows then that $\mathbf{R} = UV^t$.

The technique used to compute **R** here is similar to the one proposed by Huang *et al.*(22,35) for the case of range data. There, they use centroids of the two range data frames to factor out the translation, and then compute **R**. Using the technique outlined above, we can compute (an approximation to) the rotation between the two frames without *a priori* knowing the correspondences between them. Note that this results in four possible candidates for the rotation matrix. The reasons for this are detailed in reference (35). Although the results are better if the correct matrix can be chosen, correspondence can be established without selecting one of these, as our results indicate.(33) In this work, we assume that the ground truth rotation will be less than 90°. If this be the case, then in general only one of the rotation matrices will yield an angle of within 100°, and this can be selected as **R**.

However, such as an assumption is somewhat restricting, and the authors are currently working on schemes to be able to remove this assertion. Consider, for instance, the following. We know from our earlier discussion that

$$\hat{B}_{\sigma(i)} = \mathbf{R}\hat{A}_i \quad i = 1,\ldots,N \tag{6}$$

Summing over i and dividing by N, the number of points will give us

$$\frac{1}{N}\sum_{i=1}^{N} \hat{B}_{\sigma(i)} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{R}\hat{A}_i \quad i = 1,\ldots,N \tag{7}$$

If we let **b** and **a** be the averages of $\hat{B}_i$ and $\hat{A}_i$ respectively, then we gét from the above that

$$\mathbf{b} = \mathbf{Ra}. \tag{8}$$

R thus is a function mapping **a** to **b**. **R** should also map the eigenvectors, v_i of V to the eigenvectors u_i of U. The reason we get four candidates for the rotation matrix is that the sign and orientation of the eigenvectors are not known *a priori*. The rotation matrix could map v_i to u_i or $-u_i$. However, equation (8) provides us a way to disambiguate. Suppose **R** maps v_i to u_i. Then the angle between **a** and v_i should be roughly the same as the angle between **b** and u_i. A similar argument holds with the angle between **b** and $-u_i$ if v_i is mapped to $-u_i$. Since we can determine the sign and orientation of the eigenvectors, we can chose the exact **R**. In practice, doing this for two eigenvectors suffices, since the third is their cross product.

We have implemented this scheme and carried out many simulations. While the scheme seems to work for most values of tilt, slant and rotation, it does occasionally fail to chose the correct **R**, especially as the angle of rotation is increased. It has also been observed that the higher the values of the tilt and slant angles, the better the scheme tends to perform. We are currently engaged in modifying this scheme so that it may work for all possible values of the **R** matrix.

Let us now establish that this **R** can also serve as the "property wX" that we have talked about earlier. Consider first an analogy with human vision. Suppose that we see two frames of some object, and form some intuitive idea of the amount of rotation required. If we see the same frames with the corresponding parts covered, we are likely to still form the same opinion about the amount of rotation involved. However, if we see the frames with non corresponding parts covered, then our estimate of rotation will be different.

Claim 3. **R**, *the estimate to the rotation, is a property of type wX*

Proof. We note that if in equation (4), the same point is dropped from both LHS and RHS, then the computed **R**′ would be very close to **R**. This will not in general happen, however, if non-corresponding points are dropped from LHS and RHS. The truth of *wX* corresponds in this situation to **R**′ approximating **R**.

Rather than use some matrix norm to measure how closely a given **R**′ approximates **R**, we instead compare the angles of rotation that these matrices represent.

The angle of rotation θ is obtained from **R** as

$$\theta = \frac{180}{\pi} \times \cos^{-1}(0.5 \times (trace\mathbf{R} - 1)).$$

We can define the error in matching point i and j, ε_{ij} to be the difference in the computed angles of rotation obtained when all points are used and when the subset obtained by dropping the those two points is used. Thus, we could obtain the optimal match as one which minimized the sum of all individual point match errors. This is the classic matching problem, known to be NP Complete. We use the following greedy heuristic for matching. For each point k in the frame, we chose as its match the point l in the second frame such that

$$\varepsilon_{kl} = min_j \varepsilon_{kj}, j = (1 \cdots n).$$

In other words, we convert **R** from a property of type wX to a property of type X by asserting that it is true for the subset which leads to the smallest difference ε in angles.

So given any subset of the set of tokens from the two frames, computing **R** can decide if it contains only corresponding tokens. It follows from claim 1 that **R** can also establish correspondence between the tokens.

In presenting the mathematics above, the following has been assumed:

(1) The translational component of the motion can be neglected when computing **R**.

(2) The object(s) in question are rigid.

(3) The actions of the imaging device can be modelled by perspective projection, and its focal length is unity.

The first assumption is best satisfied if the object were some distance away from the imaging device and did not have a very large translational velocity. We emphasise once again that the object here is not to accurately compute the motion parameters, but merely use **R** as a means to establish correspondence. Thus a rough estimate to **R** suffices. Given correspondence, there is much literature on how to accurately compute motion. Moreover, we present simulation results that indicate that this assumption holds fairly well, even in the presence of non-negligible translation.(33)

Note that in the above discussion, we have assumed that there is a unique **R** which describe the motion. If the object is symmetric, then the feature points obtained from it are likely to be symmetric as well. In such cases, there may be more than one rotation that explains the point positions in the two frames. In such scenarios, the algorithm can make errors, especially when the points dropped from the two frames are non corresponding, but symmetrically placed. However, the extension to this basic algorithm that we describe in Section 4.1 to handle missing points does not suffer from this problem, as long as there are some (three or more) non symmetric points. Of course, were all the points to be symmetric (say a cube), the algorithm would get confused.

3. LINE TOKENS

While expounding the mathematical basis of the algorithm in the previous chapter, we assumed that the tokens were points. Very often however, the tokens that are extracted from image frames are the edges of the objects and as such are lines. As Aggarwal and Wang point out,(36) there seems no compelling reason to use only point tokens as opposed to line or plane tokens. They propose an algorithm which uses both line and point correspondences to establish motion and structure. Huang and Liu(37) also propose a motion estimation algorithm which uses correspondences between lines. However, attempts to establish correspondences between line tokens have been relatively few. In reference (38) the authors use parameters like distance from camera and relative movement to do line correspondences. Faugeras *et al.*(39) use an approach which involves prediction of line parameters in the next frame using Kalman filtering and then trying to match using some normalized distance measure. They represent lines with a 4D feature vector. More recently, Goldgof *et al.*(40) have proposed an algorithm that uses 3D data to obtain correspondence for lines. This is done by identifying two coordinate directions based on line positions in the two frames. The relationship between these coordinate directions is used to obtain the motion parameters, from which the correspondence is recovered. The recovery process is similar to the one described in reference (35) and hence subject to a similar sensitivity to noise. An extension of this work, which uses scaled orthographic projections of the 3D points or lines, is presented in reference (41). In contrast, our approach uses 2D data, and uses a strategy to represent lines such that our basic point correspondence algorithm can be applied. We present this approach next.

Consider any two points a and b on the lines, and the origin, o. Together, these uniquely identify a plane containing the line and passing through the origin. Taking the cross product of $\vec{oa}$ and $\vec{ob}$ will generate a vector $\vec{oc}$ which is the normal to this plane. Point c can serve as a representation of the line in question. We point out here that all lines in space that project to the given line token in the image will have c as a representation. Note that the choice of the two points is arbitrary, any two points on the line would do. This is because in our algorithm, we convert all inputs to unit vectors, as is evidenced from equation (3). Consequently, what matters is the direction of the position vector of the input point. Since this direction is the normal to the plane, it does not matter if we chose the same points in the two frames or different, as long as they lie on the line. By the same token, this method will not be able to disambiguate multiple line segments which are collinear. We now establish the following claim:

Claim 4. *If the line is rotated by* **R**, *then the corresponding point c rotates by the same amount.*

Proof. Without loss of generality, assume that the same two points are chosen on the line in the two frames. So, the position vectors in the first frame are $\vec{oa}$ and $\vec{ob}$. Since the rotation is described by **R**, the position vectors in the second frame shall be $\mathbf{R}\vec{oa}$ and $\mathbf{R}\vec{ob}$. If c' be the position of the cross product in the second frame, then

$$\vec{oc}' = \mathbf{R}\vec{oa} \times \mathbf{R}\vec{ob} = \mathbf{R}(\vec{oa} \times \vec{ob}) = \mathbf{R}\vec{oc}. \qquad \square$$

It follows from the above that point c is a valid representation of the line. This transformation is depicted in Figs. 1 and 2. In Fig. 1, we see the lines in the two frames. The motion between the first and second frames involves a rotation of 20° and a translation by 40 units along the three axes. In Fig. 2, the transformed points are shown for the lines in Fig. 1. In Table 1, we present the results of running this algorithm on synthesised data. For the ideal case of no translation, the algorithm matched all correct tokens. Even with translation between the frames, the algorithm correctly matched a large majority of the points. The results clearly demonstrate that transformation technique we suggest works. In Fig. 3 we see a real image sequence. The ground truth data here is that the camera moves a few inches leftwards and downwards, and a few feet towards the building. No rotation is involved. We were able to correctly match five out of the six line tokens used. We may point out here that the coordinates of the lines were picked out using a mouse pointer, and as such are prone to errors. However, as we showed in reference (33) the method is not significantly affected by small noises in the data values. Interestingly, since we have converted line tokens into point tokens, the input to our algorithm can in fact be a mix of point tokens as well as the point representation of line tokens. This is significant because it means that we can use both edges and corners, which are typically selected by feature detectors, as input to our algorithm. Apart from the obvious benefit of being able to use a larger number of features, there is also some evidence[42] to indicate that the presence of both edges and corners helps in the correspondence process. For the image sequence shown in Fig. 4, a combination of point and line tokens were used as input to the algorithm. These images were taken by a camera mounted on a PUMA robotic arm. The ground truth here is not available, but it is believed that the camera is moving right, down and towards the scene, and is making an angle of about 52° to the horizontal axis. Once again,

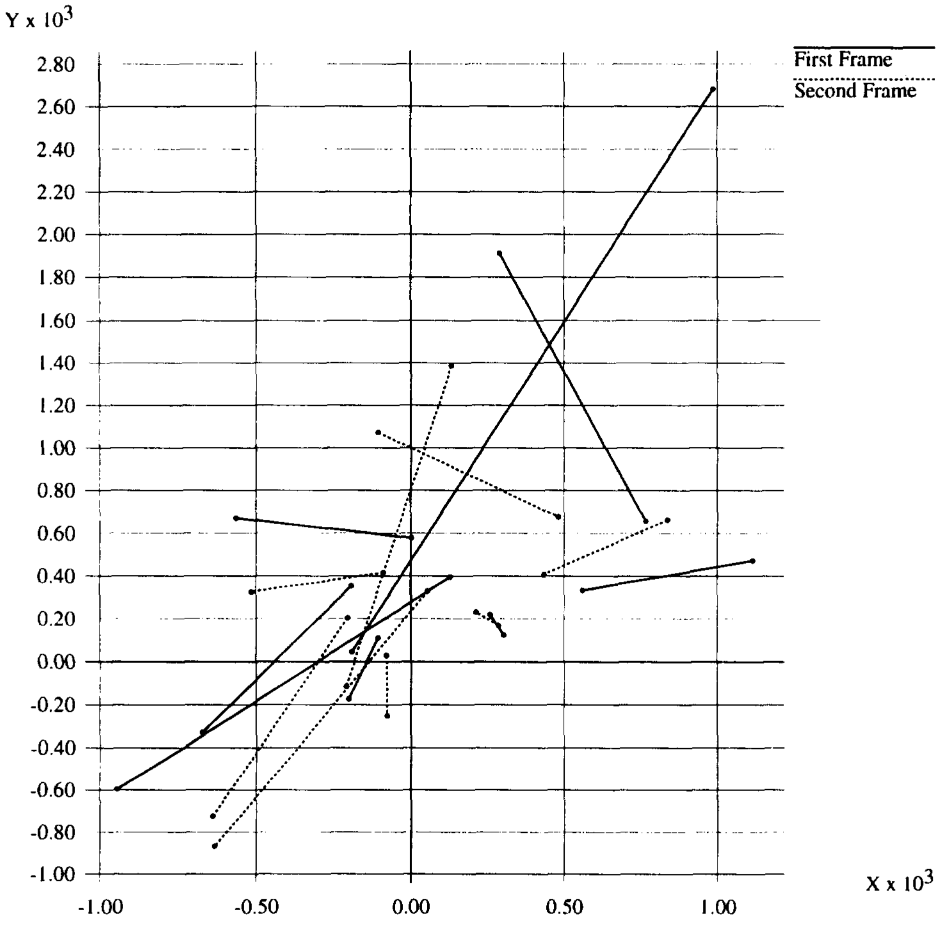

Fig. 1. Lines in the two frames.

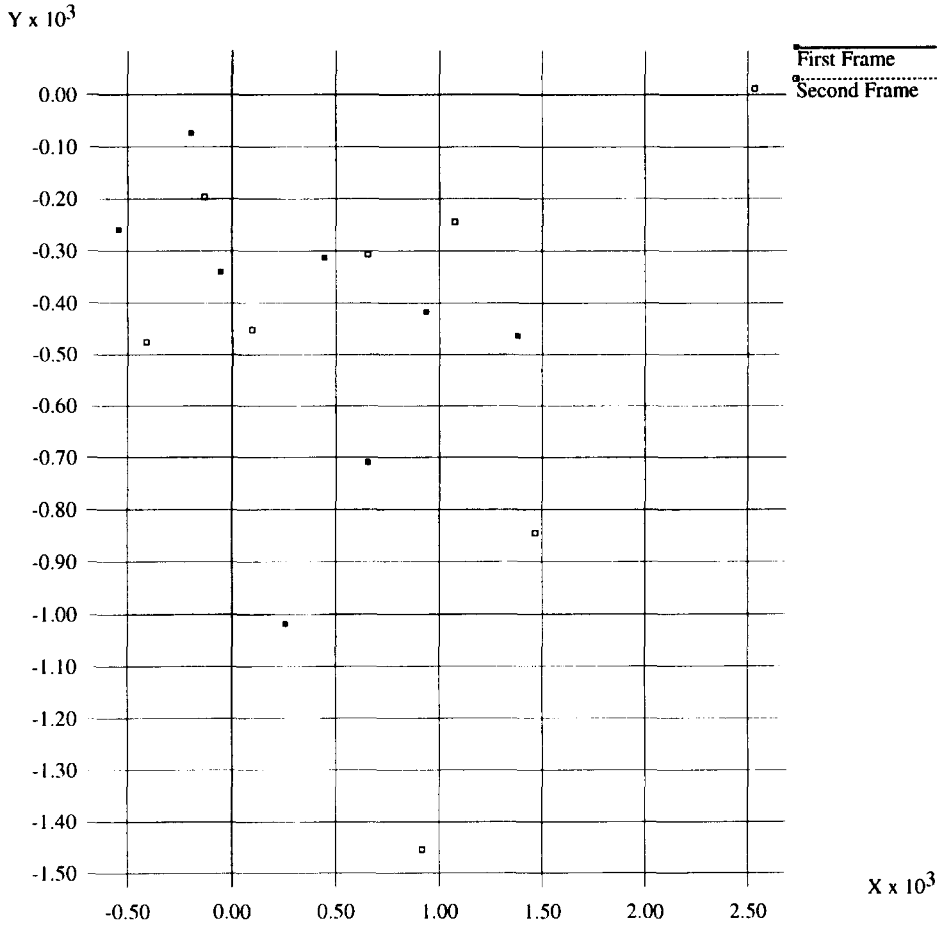

Fig. 2. The points corresponding to lines.

Table 1. Line tokens: the entries represent the number of tokens correctly matched out of seven

	Angle in degrees					
Translation	10	20	30	50	75	90
0 0 0	7	7	7	7	7	7
20 20 20	6	7	6	5	6	6
40 40 40	5	6	5	6	6	6

little or no rotation seems to be involved. In this case, we were able to match correctly eight out of the nine tokens.

4. MISSING TOKENS

In our work with the basic algorithm dealing with point tokens, as well as the extension dealing with line tokens, we have made the assumption that no tokens are missing between frames, i.e. give a token from the first frame, its corresponding token exists in the second frame. This assumption seems implicit in much of the work dealing with correspondence. In the "real world" however, this assumption rarely holds. The process of feature extraction may fail to extract the same set of features in the two frames, or the data may be very noisy for a certain token, or the feature corresponding to a token may be occluded or out of the camera's view area in the other frame. So instead of being some kind of rarity, missing tokens are in fact very likely to arise when dealing with real images. This changes the nature of the problem somewhat by creating a condition where a given token may not have a corresponding counterpart in the second frame. Our algorithm, as outlined in Section 2.2 will fail in this case. This is because we can no longer make the order of point appearances immaterial when making the transition from equation (3) to equation (4). Very little work has been done to address this problem. Huang *et al.*(23) address this problem, but they assume availability of 3D data for points. Rangarajan and Shah(26) use their proximal uniformity constraint to predict the position of missing points. However, their method requires that all points be available in the first two frames. Salari *et al.*(43) also use the continuity of motion in a similar fashion to handle missing points.

We now propose a novel and simple technique which enables our basic algorithm to handle missing tokens.

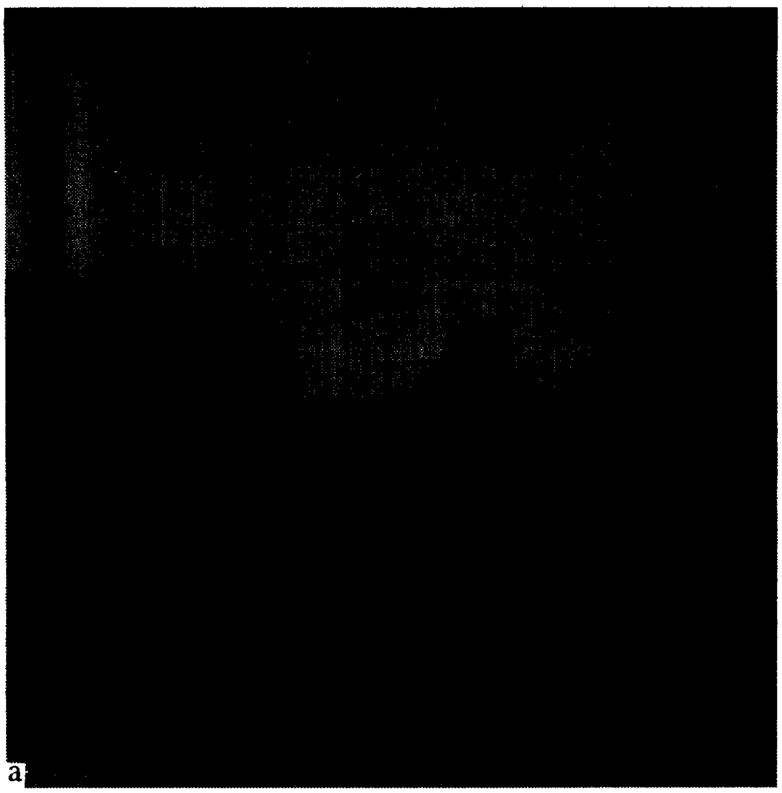

Figure 3a

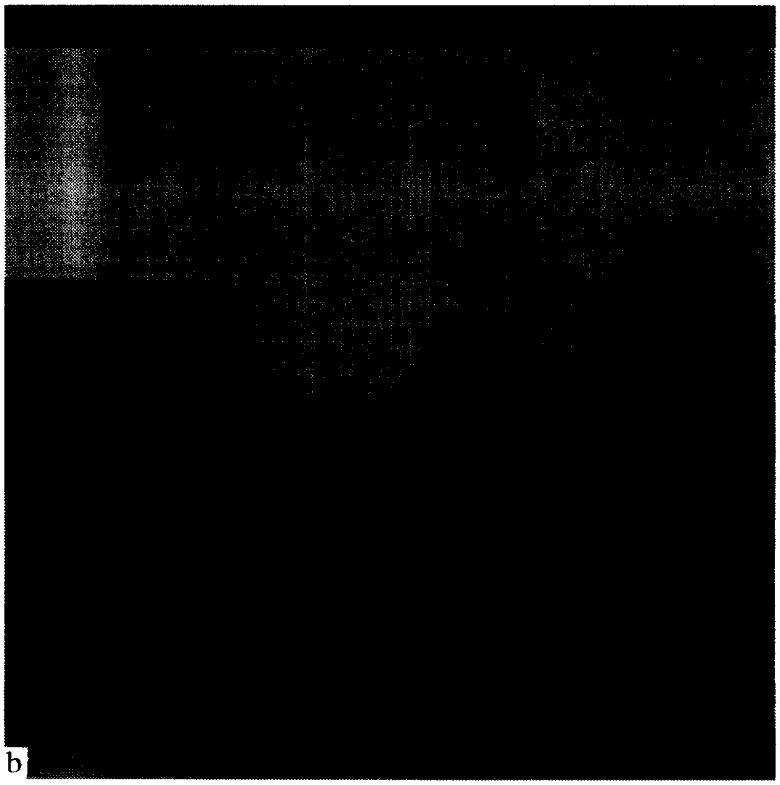

Figure 3b

Fig. 3. Two frames of an outdoor sequence.

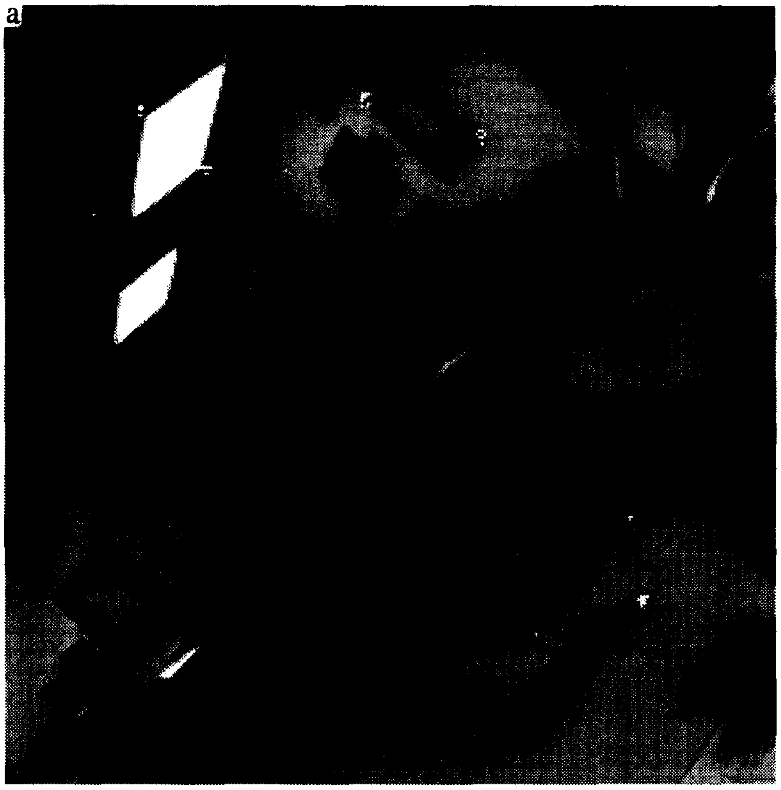

Figure 4a

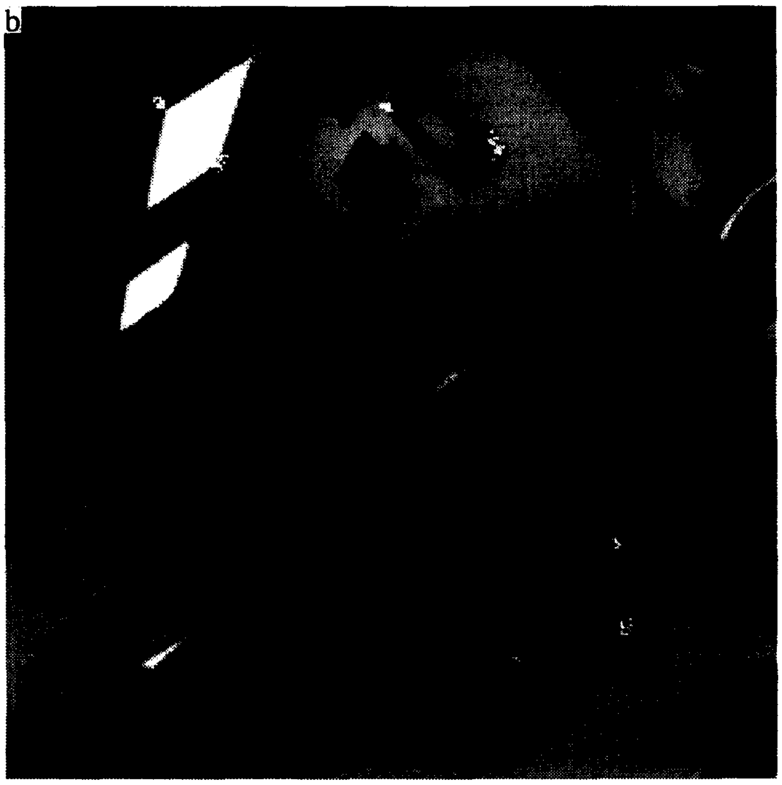

Figure 4b

Fig. 4. Images obtained from a camera mounted on a robotic arm.

4.1. *Basic idea*

To simplify the exposition of the idea, let us assume that there are an unequal number of tokens in the two frames, say n in the first and m in the second. Without loss of generality, assume that $n > m$. Note that while an unequal number of tokens in the two frames guarantees that there are missing tokens, an absence of such inequality does not imply that all corresponding tokens are present. Assuming that too many tokens are missing and one could estimate where the missing tokens should have been, one could once again use the computation of **R** to determine correspondence. We submit that such an estimate does not have to be very accurate. This is because we merely wish to correspond those points for which the corresponding tokens exist. Our aim is not to predict the position of the missing token with any accuracy.

Since we do not have any *a priori* knowledge of the object or its motion, we need to come up with some sort of a statistical scheme. We observe first that most feature detectors do pick up feature points from the object boundary. Thus it should be safe to assume that the tokens we have contains some form the boundaries of the object. Beyond this, of course, we have no knowledge about the method of feature extraction, so we assume that the feature detection mechanism is equally likely to pick up any point from the object. In other words, we argue that the points that are used as tokens can be regarded as having been randomly sampled from the object, and that they follow a uniform distribution. This enables us to devise a simple scheme to "generate" points in the second frame. We find out the maximum and minimum amongst the x and y coordinates of the m points. We then generate in the second frame, $n - m$ points. Each of these "generated points" has a x coordinate that is a random number chosen between the maximum and minimum x of the m original points. It's y coordinate is chosen similarly. We now show, that this process is the maximum likelihood estimation of the x and y coordinates of the points. We present first, a standard result about uniform distribution.[44]

Lemma 1. *Let $x_1 \cdots x_n$ be random variables from a uniform distribution $U(\alpha, \beta)$. Then $U(x_1, x_n)$ is the MLE to this distribution.*

Proof. The likelihood function in this instance is

$$L(\theta|x_1,\ldots,x_n) = \frac{1}{(\beta-\alpha)^n} I_{[\alpha,\infty)}(x_1) I_{(-\infty,\beta]}(x_n)$$

which is clearly maximised when $\beta - \alpha$ is minimum. Note that α and β are subject to the conditions $\alpha \le x_i$ and $\beta \ge x_n$. Hence the maxima occurs when $\alpha = x_1$ and $\beta = x_n$

Claim 5. *Our method for generating points is a MLE*

Proof. Follows trivially from lemma 1. Our method simply uses the MLE distribution for the X and Y coordinates.

Note that if n was the same as m, then we would not have generated any tokens.

Given the new augmented set of tokens, we once again run our correspondence algorithm on it. However, we now know that some tokens may not have their corresponding tokens present. We handle this condition as follows. Suppose that for the ith token, we compute the new rotation angle from $\mathbf{R}'$ by successively dropping points from the second frame. If we find that for no $k (k = 1 \cdots n)$ does $\mathbf{R}'$ come within some predetermined threshold of $\mathbf{R}$, then we declare that i has no match in the second frame. We empirically chose this threshold to be 10° in our case. Of course, the "generated points" are not sought to be corresponded. So the algorithm to do the match is as follows

(1) If there are no missing points, go to step 3.
(2) Using the MLE scheme presented, generate the missing points.
(3) Run the basic algorithm.
(4) If "enough" points are matched, stop. Otherwise go to step 2. □

We carried out simulations of this idea using synthesised data. In general, we observed that it would correctly match between 30–60% of the tokens when up to 15–20% were missing in the second frame. If the algorithm was repeated often enough, it would generally match 80% of the tokens correctly. Of course, the fewer the number of missing tokens, the smaller wuld be this "often enough" number.

While interesting in themselves, these results are no sufficient. For one, it takes many iterations before a significant number of correct matches are obtained. To obtain a large percentage of correct matches using this scheme will thus need an inordinate amount of time. Moreover, all the matches that the algorithm establishes are not necessarily correct. We observed that while most of the matches found by the algorithm were correct, some incorrect matches would invariably slip in. If the threshold of acceptable difference was reduced significantly to avoid such a situation, valid matches would get rejected as well. One would therefore wish to use such a method only if a better alternative was not available. In the next section, we examine how and under what conditions this performance can be improved and propose a new scheme that can obtain almost all of the correct matches.

4.2. *Can we do better: The Growing Phase*

As Aloimonos *et al.* have observed,[45] it is very often the case that some *ground truth* information regarding correspondence is available from other cues. They use such pre-established correspondences to obtain even more correspondences. In order to improve the performance of our algorithm in the presence of missing tokens, we require the satisfaction of a much weaker constraint, namely the verification of some hypothesised correspondences. We assume in what follows that given a set of hypothesised point to point correspondences, we can decide which of those are correct.

Let us now see how we can improve the performance of our algorithm. Once again, we refer back to our initial discussion in Section 2.1. Suppose we had a subset of tokens $\mathscr{S}'$ (of the form $\hat{\mathscr{S}}$), and were given two more tokens. We could decide if these tokens corresponded by simply adding them to $\mathscr{S}'$ and seeing if this augmented set was also of the type $\hat{\mathscr{S}}$. Given a set of tokens $\mathscr{P}$, and a subset of it $\mathscr{P}'$ which is of the form $\hat{\mathscr{S}}$, we can go on augmenting $\mathscr{P}'$ and depleting $\mathscr{P}$ until all the corresponding points were in the former, and the latter was left with tokens which did not have any corresponding members in the other frame. This is, in a sense, the exact reverse of *reduction*, and uses the idea that *whatever is true of the corresponding parts must be reflected in the whole*. For the lack of a more appropriate term, we call it holist.

In terms of our algorithm, this means the following. First, generate points for the frame which has the smaller number of points. Then, use the basic algorithm to obtain correspondence. Using some external "Oracle", verify which of the hypothesised correspondences are correct. **Refine** the estimate of **R** using only those token pairs which are correctly corresponded. It can be shown that as long as there are three correct matches, the process of refining **R** will come up with an accurate value for it. So in essence, all we need to demand from our estimation process is that it allows some three points to be correctly matched.

Using this core of correct matches, we can **Grow** the solution using the idea outlined in the previous paragraph. Using the analogy with human reasoning once again, imagine that we are shown two frames containing corresponding parts of the object. From these, we make an estimate of the rotation involved. Now, if these parts are augmented by uncovering more corresponding parts, then our estimate of rotation will not change. However, if we uncover non corresponding parts in the two frames, then obviously our rotation estimate will be different. Note that for such reasoning to work accurately, we must initially see a "large enough" part of the object. In procedural terms, consider an unmatched token in the first frame, say *i*. The **R**$'$ computed by adding this and another token from the second frame will be the closest to the **R** obtained by refinement when the token from the second frame is the one that corresponds to it. Thus more correspondences can be obtained from the original ones. Again, since a token is not guaranteed to have a match we set up a threshold. The computed **R**$'$ has to be within this threshold to be considered. We set this at 5° for our simulations.

The above algorithm can be outlined as follows:

(1) If there are no missing points, run the basic algorithm and stop.

(2) Using the MLE scheme presented, generate the missing points.

(3) Run the basic algorithm.

Table 2. One missing token: for each angle, the first entry is the average iterations needed and the second entry is the average number of tokens correctly matched

	Angle in degrees					
	Iterations			Matches		
Translation	30		50		70	
0 0 0	1.66	13.94	1.14	14	1	13
20 20 20	1.1	11.3	3.9	10.86	1	10.08

Table 3. Two missing tokens: for each angle, the first entry is the average iterations needed and the second entry is the average number of tokens correctly matched

	Angle in degrees					
	Iterations			Matches		
Translation	30		50		70	
0 0 0	3.1	12.94	1.6	13	1	13
20 20 20	2.1	11.12	1.7	10.86	1	10

Table 4. Four missing tokens: for each angle, the first entry is the average iterations needed and the second entry is the average number of tokens correctly matched

	Angle in degrees					
	Iterations			Matches		
Translation	30		50		70	
0 0 0	11.16	10.92	8.3	11	1.76	11
20 20 20	15.7	9.7	37.48	8.18	17.04	8.08

(4) If all points are matched, stop.

(5) From the matches proposed by the basic algorithm, select the valid ones.

(6) Using these valid matches, GROW more correct matches by incrementally adding token pairs and testing if they are a valid match.

(7) stop.

We tested this idea as usual on both synthesised data as well as real images. As the accompanying results show, it works extremely well. Tables 2, 3 and 4 show the results with the synthesised data. The original data set consists of 15 points, and we show the average number of iterations needed before three (or more) tokens were correctly matched, and the total number of points the algorithm could correctly match after the refinement and growing phases. The averages were taken over 50 runs. The intuitively obvious trend of more iterations being required if more points are missing is reflected in the data. There are, of course, minor aberrations in the trend, since we have averaged only over 50 runs.

In Figs 4 and 5, we show real image sequences on which the algorithm was tested. The ground truth data

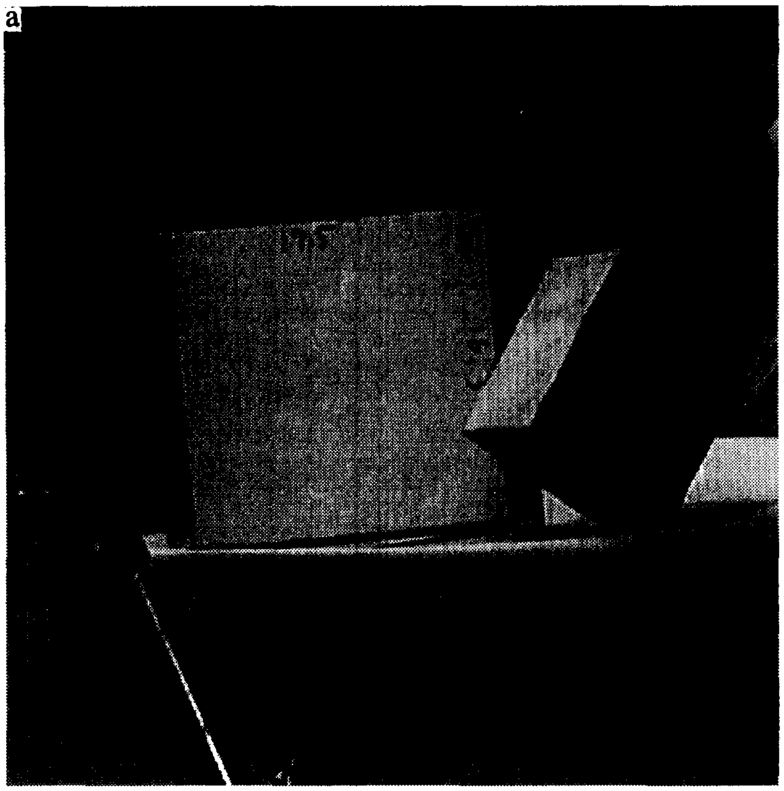

Figure 5a

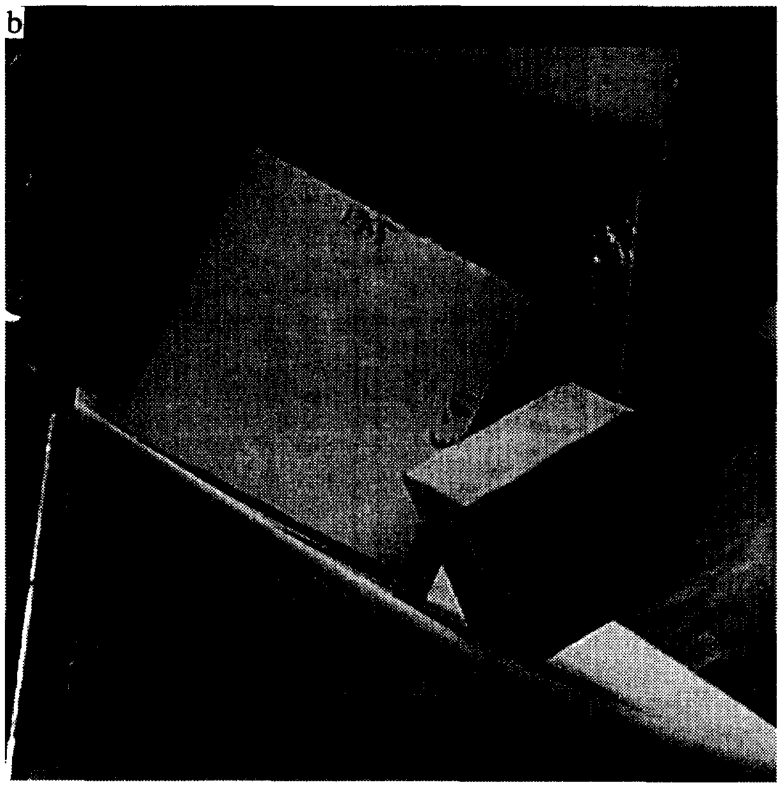

Figure 5b

Fig. 5. Image sequence taken from a tripod mounted camera.

for Fig. 4 has been discussed before. Eleven points were used from this image, and the algorithm worked fairly quickly (< 10 iterations) with up to four being dropped. For Fig. 5, the ground truth involves the camera being rotated by about 25 to 30 degrees and negligible translation. This time, the 10 lines were extracted from the figure and used as input. Once again, for up to four lines being dropped, the algorithm was fairly quick in obtaining the correct matches. In both the above cases, all tokens were matched correctly most of the time. These results clearly demonstrate the success of the algorithm.

While outlining the idea to handle missing tokens (Section 4.1), we have been vague at two points. Firstly, we said "assuming not too many points are missing". After running extensive simulations, we say with some confidence that up to 25% of points can be missing for the algorithm to not take inordinately long to obtain correspondence. Even if more points were missing the algorithm should, given "sufficient" iterations, manage to correctly match three points, thence managing to grow the solution. In general, however, as the trends from the experimental data clearly show, the number of iterations needed seems to grow quite fast as more points are missing. Secondly, we said "such an estimate does not have to be very accurate" when talking about generating tokens. The question naturally arises as to how accurate does this estimate have to be. In reference (26), the authors use information from frames k and $k-1$, as well as the smoothness constrains to generate the missing point in frame $k+1$. Our simulation results clearly show that even a model as simple and naive as uniformly distributed random points serves the purpose of establishing correspondence. This, we feel, is eloquent testimony to the ideas of *refinement* and *growing*.

5. DISCUSSION

Feature based schemes for image sequence analysis have for long assumed that the underlying problem of correspondence has been resolved. Most algorithms to establish correspondence have, however, operated under restrictive assumption. In this work, the authors have presented an improvement to their previously proposed algorithm which enables it to use both line as well as point tokens, and even handle the case of missing tokens. It is also fairly immune to noise in the input data. This makes the proposed algorithm suitable for handling tokens obtained from feature extractors operating on real images.

The basic ideas underlying the algorithms are inspired by the way human thinking seems to work about the motion of an object, viz., *whatever is true of the whole must be true for its parts and whatever is true of a part must be reflected in the whole.* The algorithm operates only for rigid bodies, and assumes that the effect of translation can be ignored when computing the rotation matrix for the purpose of correspondence. We have presented arguments to justify this not so obvious assumption and show its validity. We have also presented extensive simulation results, both on synthesised data, as well as on real images, that not just establish the veracity of our algorithm, but also *a posteriori* justify our assumptions.

However, the ability of this method to establish correspondence correctly does not detract from our belief that in any real visual system, correspondence must be established using a variety of cues. We submit that a motion based approach like ours provides one of the most important cues that aid in this process.

A word on the computational complexity of the algorithm. As was noted in reference (33), the basic algorithm requires $O(n^2)$ computations of $\mathbf{R}$ to obtain correspondence. The process of converting line tokens into their point representation is linear in the number of tokens, and so does not add to the complexity. Nor does the process of growing, which also needs $O(n^2)$ computations of $\mathbf{R}$.

The way the algorithm is structured to handle missing tokens, it can be expanded to deal with a case of correspondence when there are two (or more) objects exercising different motion, assuming some not too restrictive assumptions can be made about the image sequence. The authors are currently engaged in developing this idea. Also, since the algorithm can establish correspondence between two frames, and does not require multiple frames, it can also be used to do image to model correspondences. In the basic algorithm, as well as in the growing phase, the computations (of $\mathbf{R}$) to establish correspondence can be done in parallel for all points. So also the process of converting line tokens to point tokens. Thus there is a high degree of inherent parallelism in the algorithm which can be exploited fairly easily.

In conclusion, we present an efficient algorithm to do correspondence in an image sequence that can handle both point and line tokens and works well even if tokens are missing. These features make the algorithm, which has an inherent parallel structure, suitable for analysis of real images.

Acknowledgements—The authors would like to thank the anonymous referee for bringing some new literature in this area to our attention and offering suggestions which improved the presentation of this paper.

REFERENCES

1. J. W. Roach and J. K. Aggarwal, Determining the movement of objects from a sequence of images. *IEEE Trans. Pattern Anal. Mach. Intell.* **2**, 29–46, (November 1979).
2. H. C. Longuet-Higgens, A computer algorithm for reconstructing a scene from two projections. *Nature*, **293**, 133–135 (1981).
3. R. Y. Tsai and T. S. Huang, Uniqueness and estimation of three dimensional motion parameters of rigid objects with curved surfaces, *IEEE Trans. Pattern Anal. Mach. Intell.* **6**, 13–27 (1984).
4. H. H. Nagel, Image sequences—ten(octal) years—from phenomenology towards a theoretical foundation, *Proc. Int. Conf. Pattern Recognition*, 1174–1185 (1986).

5. J. K. Aggarwal, Motion and time varying imagery—an overview, *Proceedings of the IEEE Workshop Motion: Representation and Analysis*, pp. 1–6 (1986).
6. J. Aloimonos, Perception of structure from motion, *Proc. Conf. Comput. Vision Pattern Recognition* (1986).
7. G. S. Young and Rama Chellappa, 3-D motion estimation using a sequence of noisy images, *Proc. Conf. Comput. Vis. Pattern Recognition* (1988).
8. J. Weng *et al.*, 3D motion estimation, understanding and prediction from noisy image sequences, *IEEE Trans. Pattern Anal. Mach. Intell.* **9**, 370–389 (1987).
9. O. D. Faugeras and S. Maybank, Motion from point matches: multiplicity of solutions, *Proc. IEEE Workshop on Motion* (1989).
10. B. K. P. Horn, Relative orientation, *Int. J. Comput. Vision* **4**, 59–78 (1990).
11. B. K. P. Horn, Recovering baseline and orientation from essential matrix, Technical report, MIT AI Memo (1990).
12. W. Burger and B. Bhanu, Estimation 3-D motion from perspective image sequences, *IEEE Trans. Pattern Anal. Mach. Intell.* **12**, 1040–1058 (1990).
13. Ken-Ichi Kanatani, Structure from motion without correspondence: general principle, Technical Report CAR-TR-164, Center for Automation Research, University of Maryland (1985).
14. J. Aloimonos and A. Basu, Shape from 3-D motion for contour without point to point correspondence, *Proc. Conf. Comput. Vision Pattern Recognition* pp. 518–527 (1986).
15. J. Aloimonos and I. Rigoutsos, Determining 3-D motion of a rigid surface patch without correspondences under perspective projection, *Proceedings AAAI*, 681–688 (1986).
16. V. Salari and I. K. Sethi, Feature point correspondence in presence of occlusion, *IEEE Trans. Pattern Anal. Mach. Intell.* **12**, 87–91 (1990).
17. R. Jain and I. K. Sethi, Finding trajectories of feature points in a monocular image sequence, *IEEE Trans. Pattern Anal. Mach. Intell.* **9**, 56–73 (1987).
18. R. Nevatia, Depth measurement from motion stereo, *Computer Vision, Graphics Image Process.* **9**, 203–214 (1976).
19. T. D. Williams, Depth from camera motion in a real world scene, *IEEE Trans. Pattern Anal. Mach. Intell.* **2**, 511–516 (1980).
20. H. Baker, *Depth from Edge and Intensity Based Stereo*, Ph.D. thesis, Computer Science Department, Stanford University (1981).
21. Y. G. Leclerc and S. W. Zucker, The local structure of image discontinuities in one dimension, *IEEE Trans. Pattern Anal. Mach. Intell.* **9**, 341–355 (1987).
22. Z. C. Lin, H. Lee and T. S. Huang, Finding 3-D point correspondences in motion estimation, *Proc. Int. Conf. Pattern Recognition* 303–305 (1986).
23. H. H. Chen and T. S. Huang, Maximal matching of two 3-D point sets, *Proc. Int. Conf. Pattern Recognition* 1048–1050 (1986).
24. D. Zhang, Perspective invariant description of a planar point set and its application to matching, *Proc. Int. Conf. Pattern Recognition* (1986).
25. W. E. L. Grimson and T. Lozano-Perez, Model based recognition and localisation from sparse range or tactile data, *International Robotics Research J.* **3**, 3–35 (1984).
26. K. Rangarajan and M. Shah, Estimating motion correspondence, *Proc. Conf. Comput. Vision Pattern Recognition* (1991).
27. J. Weng, N. Ahuja and T. Huang, Matching Two Perspective Views, *IEEE Trans. Pattern Anal. Mach. Intell.* **14**, 806–825 (1992).
28. Y. Liu and T. Huang, Vehicle type motion estimation from two image sequences. *IEEE Trans. Pattern Anal. Mach. Intell.* **15**, 802–807 (1993).
29. J. Weng, T. Huang and N. Ahuja, Motion and structure from line correspondences: closed-form solution, uniqueness and optimization, *IEEE Trans. Pattern Anal. Mach. Intell.* **14**, 318–336 (1992).
30. J. Weng, Windowed fourier phase: Completeness and signal reconstruction, *IEEE Trans. Signal Process.* **41**, 657–666 (1993).
31. A. Joshi and C. H. Lee, On the problem of correspondence in range data and some inelastic uses for elastic nets, *IEEE Trans. Neural Networks*, **6**, 716–723 (1995).
32. A. Joshi and C. H. Lee, Using elastic nets for correspondence in range data, *Proceedings ICNN'93*, San Francisco (1993).
33. C. H. Lee and A. Joshi, Correspondence problem in image sequence analysis, *Pattern Recognition* **26**, 47–61 (1993).
34. S. Ullman, *The Interpretation of Visual Motion*. MIT Press, Cambridge, MA (1979).
35. Z. C. Lin and T. S. Huang *et al.*, Motion estimation from 3-D points sets with and without correspondence *Proceedings Conference on Computer Vision and Pattern Recognition*, pp. 194–201 (1986).
36. J. K. Aggarwal and Y. F. Wang, Analysis of sequence of images using point and line correspondences, *Proceedings of the IEEE Workshop on Motion: Representation and Analysis*, pp. 1275–1279 (1986).
37. Y. Liu and T. S. Huang, A linear algorithm for motion estimation using straight line correspondences, *Computer Vision, Graphics Image Process.* **44**, 35–57 (1988).
38. J. H. McIntosh and K. M. Mutch, Matching straight lines. *Computer Vision, Graphics Image Process.* **43**, 386–408 (1988).
39. O. Faugeras and R. Deriche, Tracking line segments, *Image Vision Comput.* **8**, 261–270 (November 1990).
40. D. Goldgof, H. Lee and T. Huang, Matching and Motion estimation of Three Dimensional Point and Line Sets Using Eigenstructure without Correspondences, *Pattern Recognition* **25**, 271–286 (1992).
41. C. Lin, D. Goldgof and W. Huang, Motion estimation from scaled orthographic projections without correspondences, *Image Vision Comput.* **12**, 95–108 (1994).
42. B. G. Kaiser and T. C. Pong, Hierarchical approach to correspondence, *IEEE Trans. Systems, Man Cybern.* **19**, 217–276 (1989).
43. V. Salari, I. K. Sethi and S. Vemuri, Feature point matching in image sequences, *Pattern Recognition Lett.* **7**, 113–121 (February 1988).
44. G. Roussas, *a First Course in Mathematical Statistics*. Addison-Wesley Reading, MA (1973).
45. J. Aloimonos and B. Kamagar-Parsi, Correspondence from correspondence, *Topical Machine Vision*, Optical Society of America (1987).

About the Author—CHIA-HOANG LEE received his Ph.D. degree from the University of Maryland, College Park, in 1983. From 1984 to 1985, he was a faculty member at the University of Maryland, Baltimore County. From 1985 to 1992, he was with the Department of Computer Science, Purdue University. He is currently the chairperson of the Department of Computer Science and Information Science at the National Chiao Tung University, Taiwan. His interests include computational vision and user interfaces.

About the Author—ANUPAM JOSHI was born in New Delhi, India. He received a Bachelor of Technology degree in electrical engineering from the Indian Institute of Technology, Delhi in 1989, and Ph.D. in computer science from Purdue University in 1993. He is currently a Visiting Assistant Professor in the Department of Computer Sciences at Purdue. His research interests include computational vision, computational intelligence, mobile computing and intelligent agents and scientific computing. He is a member of IEEE, ACM and $\gamma\Pi$E.